《实验一》

上机实验指导书

**(最新版)**

2024年10月

# 实验一：动态与静态逆向工具的使用

**实验项目名称：动态与静态逆向工具的使用**

**实验项目性质：验证**

**所属课程名称：逆向工程技术**

**实验计划学时：4学时**

**一、实验目的**

掌握逆向工具OllyDBG软件的使用，了解IDA Pro的使用。

**二、实验内容**

1.认真阅读和掌握本实验的相关知识点；

2.完成实验并将实验结果记录到实验报告中。

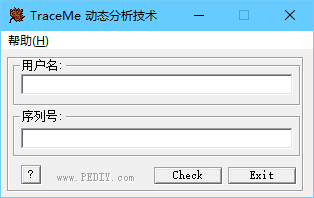
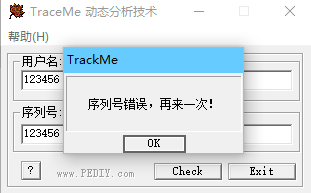
**三、实验内容**

1. 上机完全操作一次课件“02\_逆向工具的使用\_动态调试.ppt”中的内容。（涉及到的exe参见实验相关的资料）
2. 使用IDA Pro反汇编以下两个程序（功能均为调用MessageBox函数弹出一个对话框）：
   1. 直接使用汇编语言编写的程序：exe程序与源代码参见实验相关的资料。要求对汇编产生的每一句汇编代码做上注释。
   2. 使用VC6.0编写的程序（以debug模式编译）：工程文件参见实验相关的资料。请认真分析并注释地址0x0040121D之前的汇编代码（其中WinMain的启动函数部分可以参见教材102页/71页。程序执行的入口点是地址0x004010C0。只需注释汇编指令。因指令较多，不用每一条指令均做注释）。
3. 分别按下面的要求使用OllyDBG修改VC6.0编译出的弹出对话框的程序：
   1. 让其无法弹出对话框。（要求保存修改以后得到的exe文件）
   2. 修改对话框中的标题和对话框内的内容，让标题演示自己的学号并让对话框内的内容显示自己姓名的拼音；（要求保存修改以后得到的exe文件）

**四、实验步骤**

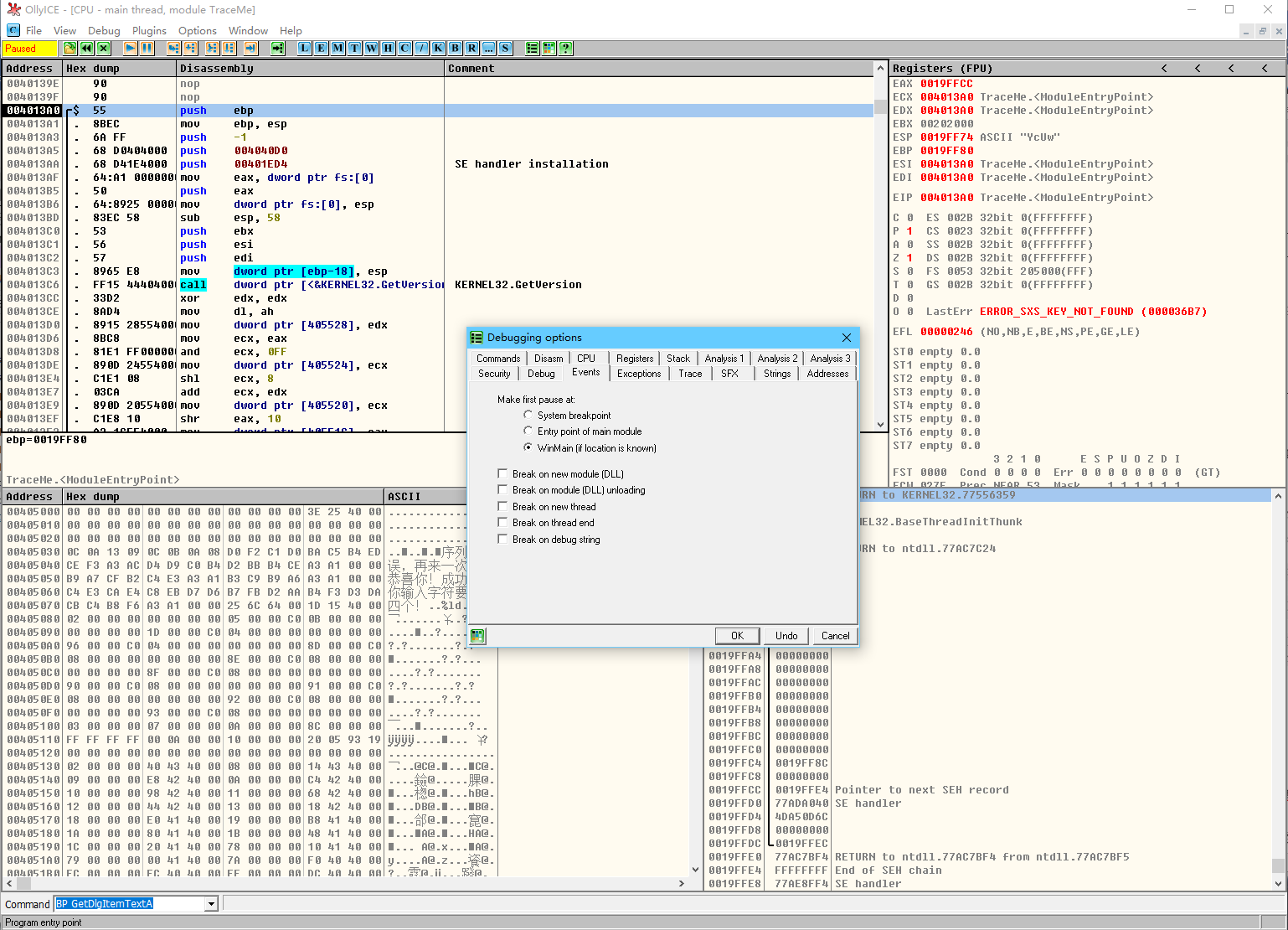
1. 上机完全操作一次课件“02\_逆向工具的使用\_动态调试.ppt”中的内容。（涉及到的exe参见实验相关的资料）

1.1、准备工作：以一个Visual C++6.0的程序TraceMe来讲解OllyDbg的操作

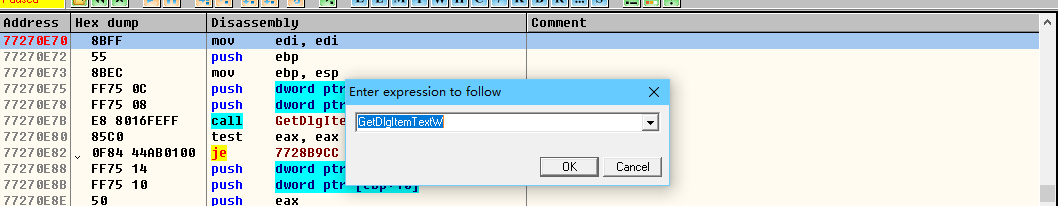
 



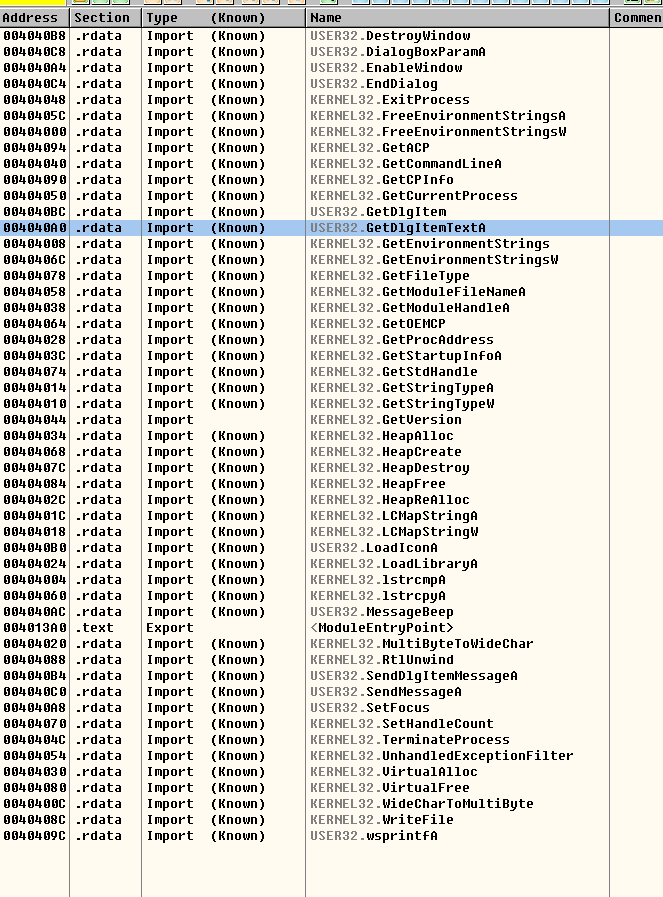
1.2、加载目标文件调试



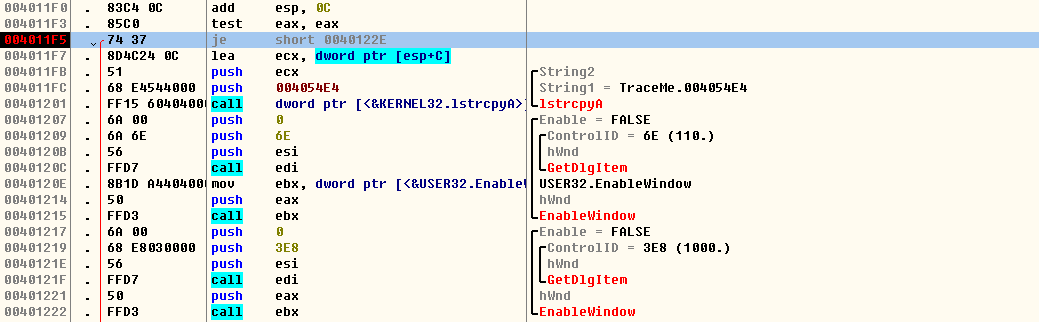
1.3、设置断点：使用“Ctrl+G“键打开跟随表达式的窗口，输入GetDlgItemTextW来，然后再下断点。



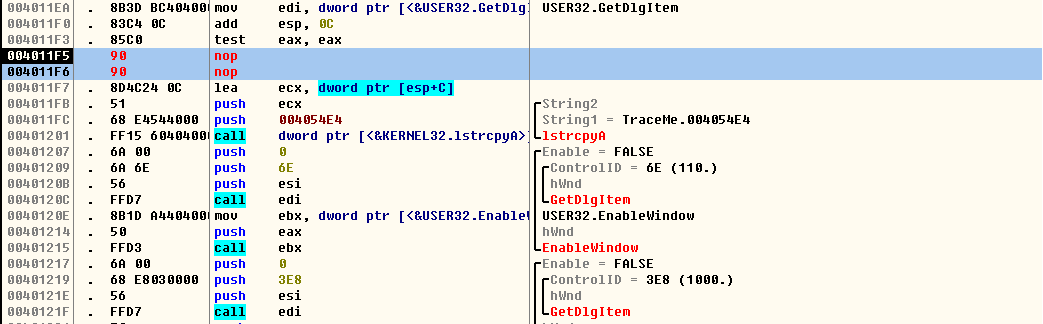
或者：按下"Ctrl+N"这个快捷键打开应用程序输入表，会发现应用程序调用的其他模块的导出函数，找到相应的函数后，通过Enter键即可切换到相应的代码，接下来按F2设置断点。



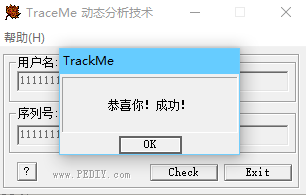
1.4、调试分析：在本例中，关键的比较代码位置为：0x004011F5，找到这个位置，将该位置的比较去掉，然后保存修改后文件即可。修改前：



在该位置按下字母E，然后输入汇编指令：NOP(空指令)即可完成破解，然后修改保存文件即可。修改后：

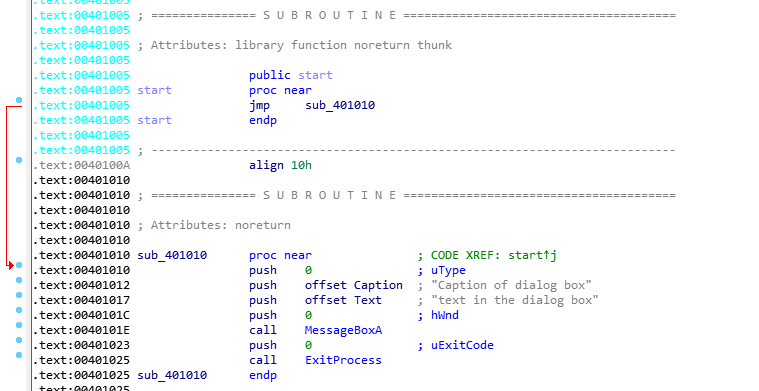


1.5、运行结果



2. 使用IDA Pro反汇编以下两个程序（功能均为调用MessageBox函数弹出一个对话框）：

a) 直接使用汇编语言编写的程序：exe程序与源代码参见实验相关的资料。要求对汇编产生的每一句汇编代码做上注释。



.**text:00401005 public start ;程序开始**

**.text:00401005 start proc near ;过程start开始**

**.text:00401005 jmp sub\_401010 ;跳转到子过程sub\_401010**

**.text:00401005 start endp ;过程start开始**

**.text:00401010 sub\_401010 proc near ;过程sub\_401010开始**

**.text:00401010 push 0 ; 将实参0入栈，传给uType**

**.text:00401012 push offset Caption ; 将字符串Caption （"Caption of dialog box"）的地址入栈**

**.text:00401017 push offset Text ; 将字符串Text （"text in the dialog box"）的地址入栈**

**.text:0040101C push 0 ; 将实参0入栈，传给hWnd**

**.text:0040101E call MessageBoxA ; 调用API MessageBoxA，弹出消息框**

**.text:00401023 push 0 ; uExitCode**

**.text:00401025 call ExitProcess ; 调用API ExitProcess，退出过程**

**.text:00401025 sub\_401010 endp ; 过程sub\_401010开始**

b) 使用VC6.0编写的程序（以debug模式编译）：工程文件参见实验相关的资料。请认真分析并注释地址0x0040121D之前的汇编代码（其中WinMain的启动函数部分可以参见教材102页/71页。程序执行的入口点是地址0x004010C0。只需注释汇编指令。因指令较多，不用每一条指令均做注释）。

程序执行的入口点：

**.text:004010C0 ; =============== S U B R O U T I N E =======================================**

**.text:004010C0**

**.text:004010C0 ; Attributes: library function noreturn bp-based frame**

**.text:004010C0**

**.text:004010C0 public start**

**.text:004010C0 start proc near**

**.text:004010C0**

**.text:004010C0 var\_20 = dword ptr -20h**

**.text:004010C0 var\_1C = dword ptr -1Ch**

**.text:004010C0 ms\_exc = CPPEH\_RECORD ptr -18h**

**.text:004010C0**

**.text:004010C0 ; \_\_unwind { // \_\_except\_handler3 ;下面为启动函数**

**.text:004010C0 push ebp**

**.text:004010C1 mov ebp, esp**

**.text:004010C3 push 0FFFFFFFFh**

**.text:004010C5 push offset stru\_422138**

**.text:004010CA push offset \_\_except\_handler3**

**.text:004010CF mov eax, large fs:0**

**.text:004010D5 push eax**

**.text:004010D6 mov large fs:0, esp**

**.text:004010DD add esp, 0FFFFFFF0h**

**.text:004010E0 push ebx**

**.text:004010E1 push esi**

**.text:004010E2 push edi**

**.text:004010E3 mov [ebp+ms\_exc.old\_esp], esp**

**.text:004010E6 call ds:GetVersion ;确定Windows版本**

**.text:004010EC mov dword\_427C6C, eax**

**.text:004010F1 mov eax, dword\_427C6C**

**.text:004010F6 shr eax, 8**

**.text:004010F9 and eax, 0FFh**

**.text:004010FE mov dword\_427C78, eax**

**.text:00401103 mov ecx, dword\_427C6C**

**.text:00401109 and ecx, 0FFh**

**.text:0040110F mov dword\_427C74, ecx**

**.text:00401115 mov edx, dword\_427C74**

**.text:0040111B shl edx, 8**

**.text:0040111E add edx, dword\_427C78**

**.text:00401124 mov dword\_427C70, edx**

**.text:0040112A mov eax, dword\_427C6C**

**.text:0040112F shr eax, 10h**

**.text:00401132 and eax, 0FFFFh**

**.text:00401137 mov dword\_427C6C, eax**

**.text:0040113C push 0**

**.text:0040113E call sub\_402C60**

**.text:00401143 add esp, 4**

**.text:00401146 test eax, eax**

**.text:00401148 jnz short loc\_401154**

**.text:0040114A push 1Ch**

**.text:0040114C call \_fast\_error\_exit**

**.text:00401151 ; ---------------------------------------------------------------------------**

**.text:00401151 add esp, 4**

**.text:00401154**

**.text:00401154 loc\_401154: ; CODE XREF: start+88↑j**

**.text:00401154 ; \_\_try { // \_\_except at loc\_4011D3**

**.text:00401154 mov [ebp+ms\_exc.registration.TryLevel], 0**

**.text:0040115B call \_\_ioinit**

**.text:00401160 call ds:GetCommandLineA ;指向进程的完整命令行的指针**

**.text:00401166 mov dword\_4295EC, eax**

**.text:0040116B call \_\_\_crtGetEnvironmentStringsA ;指向环境**

**.text:00401170 mov dword\_427C50, eax**

**.text:00401175 call \_\_setargv**

**.text:0040117A call \_\_setenvp**

**.text:0040117F call \_\_cinit**

**.text:00401184 mov ecx, envp**

**.text:0040118A mov dword\_427C8C, ecx**

**.text:00401190 mov edx, envp**

**.text:00401196 push edx ; 将envp参数压栈**

**.text:00401197 mov eax, argv**

**.text:0040119C push eax ; 将argv参数压栈**

**.text:0040119D mov ecx, argc**

**.text:004011A3 push ecx ; 将argc参数压栈**

**.text:004011A4 call \_main ;调用主函数**

**.text:004011A9 add esp, 0Ch**

**.text:004011AC mov [ebp+var\_1C], eax**

**.text:004011AF mov edx, [ebp+var\_1C]**

**.text:004011B2 push edx ; int**

**.text:004011B3 call \_exit ;退出程序**

**.text:004011B8 ; ---------------------------------------------------------------------------**

**.text:004011B8**

**.text:004011B8 loc\_4011B8: ; DATA XREF: .rdata:stru\_422138↓o**

**.text:004011B8 ; \_\_except filter // owned by 401154**

**.text:004011B8 mov eax, [ebp+ms\_exc.exc\_ptr]**

**.text:004011BB mov ecx, [eax]**

**.text:004011BD mov edx, [ecx]**

**.text:004011BF mov [ebp+var\_20], edx**

**.text:004011C2 mov eax, [ebp+ms\_exc.exc\_ptr]**

**.text:004011C5 push eax ; ExceptionInfo**

**.text:004011C6 mov ecx, [ebp+var\_20]**

**.text:004011C9 push ecx ; int**

**.text:004011CA call \_\_XcptFilter**

**.text:004011CA ; ---------------------------------------------------------------------------**

**.text:004011CF db 83h**

**.text:004011D0 ; ---------------------------------------------------------------------------**

**.text:004011D0 les ecx, [eax]**

**.text:004011D2 retn**

**.text:004011D3 ; ---------------------------------------------------------------------------**

**.text:004011D3**

**.text:004011D3 loc\_4011D3: ; DATA XREF: .rdata:stru\_422138↓o**

**.text:004011D3 ; \_\_except(loc\_4011B8) // owned by 401154**

**.text:004011D3 mov esp, [ebp+ms\_exc.old\_esp]**

**.text:004011D6 mov edx, [ebp+var\_20]**

**.text:004011D9 push edx ; int**

**.text:004011DA call \_\_exit**

**.text:004011DA ; } // starts at 401154**

**.text:004011DA ; } // starts at 4010C0**

**.text:004011DA start endp**

**.text:004011DA**

**.text:004011DF ; ---------------------------------------------------------------------------**

**.text:004011DF mov ecx, [ebp-10h]**

**.text:004011E2 mov large fs:0, ecx**

**.text:004011E9 pop edi**

**.text:004011EA pop esi**

**.text:004011EB pop ebx**

**.text:004011EC mov esp, ebp**

**.text:004011EE pop ebp**

**.text:004011EF retn**

**.text:004011F0**

**.text:004011F0 ; =============== S U B R O U T I N E =======================================**

**.text:004011F0**

**.text:004011F0 ; Attributes: library function noreturn bp-based frame**

**.text:004011F0**

**.text:004011F0 \_\_amsg\_exit proc near ; CODE XREF: \_\_setenvp+8C↓p**

**.text:004011F0 ; \_\_setenvp+F2↓p ...**

**.text:004011F0**

**.text:004011F0 arg\_0 = dword ptr 8**

**.text:004011F0**

**.text:004011F0 push ebp**

**.text:004011F1 mov ebp, esp**

**.text:004011F3 cmp dword\_427C58, 2**

**.text:004011FA jz short loc\_401201**

**.text:004011FC call \_\_FF\_MSGBANNER**

**.text:00401201**

**.text:00401201 loc\_401201: ; CODE XREF: \_\_amsg\_exit+A↑j**

**.text:00401201 mov eax, [ebp+arg\_0]**

**.text:00401204 push eax**

**.text:00401205 call \_\_NMSG\_WRITE**

**.text:0040120A add esp, 4**

**.text:0040120D push 0FFh ; int**

**.text:00401212 call off\_424A30**

**.text:00401212 \_\_amsg\_exit endp**

**.text:00401212**

**.text:00401218 ; ---------------------------------------------------------------------------**

**.text:00401218 add esp, 4**

**.text:0040121B pop ebp**

**.text:0040121C retn**

**.text:0040121C ; ---------------------------------------------------------------------------**

**.text:0040121D align 10h**

下面是主函数的内容：

**.text:00401005 \_main proc near ; CODE XREF: start+E4↓p**

**.text:00401005 jmp \_main\_0**

**.text:00401005 \_main endp**

**.text:00401005**

**.text:00401005 ; ---------------------------------------------------------------------------**

**.text:0040100A align 10h**

**.text:00401010**

**.text:00401010 ; =============== S U B R O U T I N E =======================================**

**.text:00401010**

**.text:00401010 ; Attributes: bp-based frame**

**.text:00401010**

**.text:00401010 \_main\_0 proc near ; CODE XREF: \_main↑j**

**.text:00401010**

**.text:00401010 var\_40 = byte ptr -40h ;声明变量var\_40**

**.text:00401010**

**.text:00401010 push ebp ;保存ebp的值**

**.text:00401011 mov ebp, esp**

**.text:00401013 sub esp, 40h ;创建大小为40h的堆栈空间**

**.text:00401016 push ebx**

**.text:00401017 push esi**

**.text:00401018 push edi ;三个寄存器压栈，进行保存**

**.text:00401019 lea edi, [ebp+var\_40] ; 将变量的值保存到edi**

**.text:0040101C mov ecx, 10h ; 循环次数10**

**.text:00401021 mov eax, 0CCCCCCCCh ; int 3中断**

**.text:00401026 rep stosd ; 循环ecx将eax的内容填入edi寄存器**

**.text:00401028 mov esi, esp ; 保存栈顶指针**

**.text:0040102A push 0 ; 传递参数uType**

**.text:0040102C push offset Caption ; 传递参数"title of Dialog box"**

**.text:00401031 push offset Text ; 传递参数 "content of dialog box"**

**.text:00401036 push 0 ; 传递参数hWnd**

**.text:00401038 call ds:MessageBoxA ;调用函数MessageBoxA**

**.text:0040103E cmp esi, esp**

**.text:00401040 call \_\_chkesp ;检查堆栈平衡**

**.text:00401045 mov eax, 1**

**.text:0040104A pop edi**

**.text:0040104B pop esi**

**.text:0040104C pop ebx ;三个寄存器出栈，进行恢复**

**.text:0040104D add esp, 40h ;回收大小为40h的堆栈空间**

**.text:00401050 cmp ebp, esp**

**.text:00401052 call \_\_chkesp ;检查堆栈平衡**

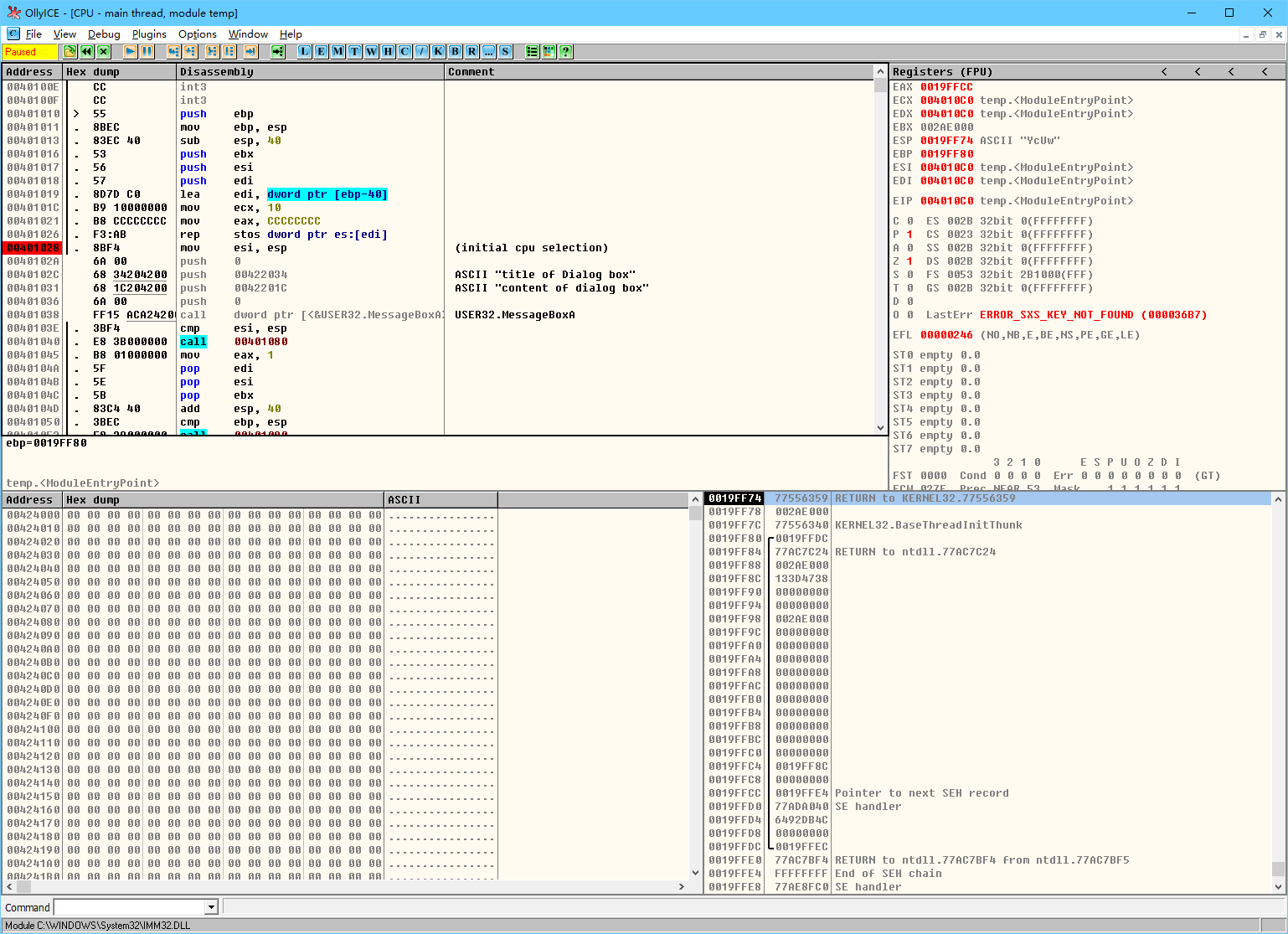
**.text:00401057 mov esp, ebp**

**.text:00401059 pop ebp ;恢复ebp的值**

**.text:0040105A retn ; 函数返回**

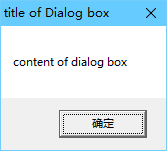
**.text:0040105A \_main\_0 endp**

3. 分别按下面的要求使用OllyDBG修改VC6.0编译出的弹出对话框的程序：

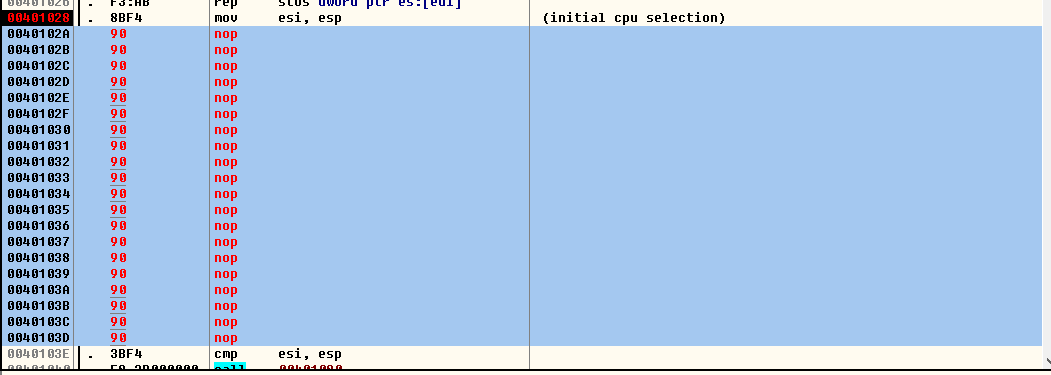


a) 让其无法弹出对话框。（要求保存修改以后得到的exe文件）

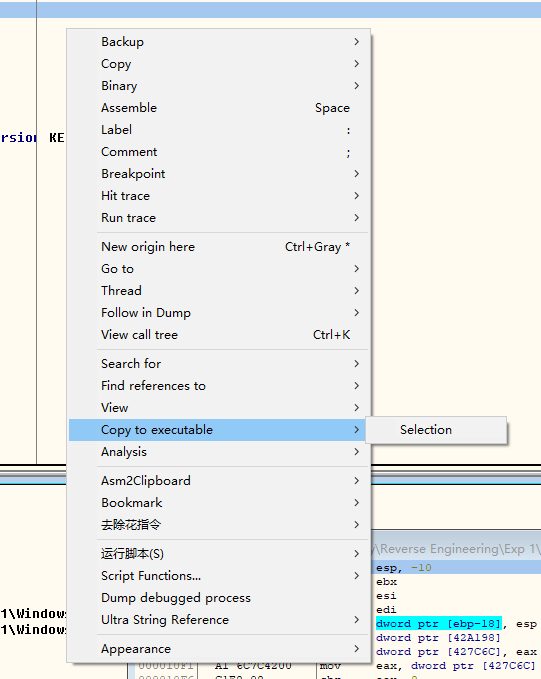
用OD打开文件，经过下断点后调试发现其在0040102A地址位置调用了MessageBoxA函数，在单步调试经过这段代码时，系统弹出了消息框



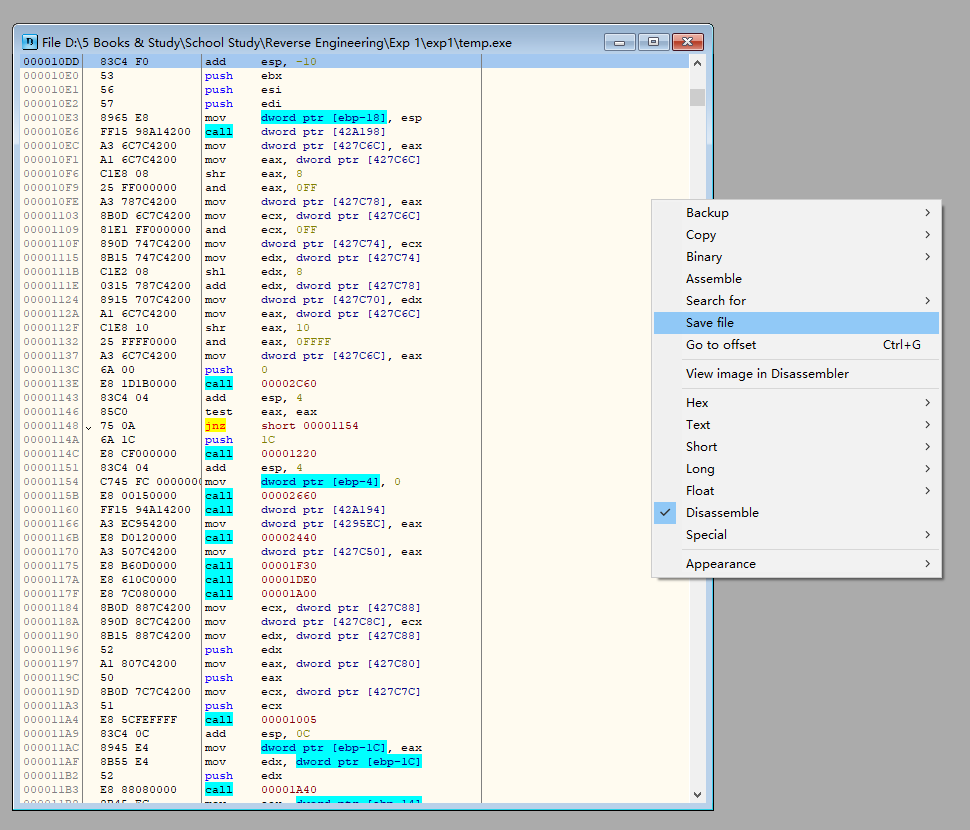
于是把这段代码NOP掉：



再次运行，发现程序没有弹出消息框，也没有异常报错，说明修改成功。右键点击修改成功的程序，复制到可执行文件：

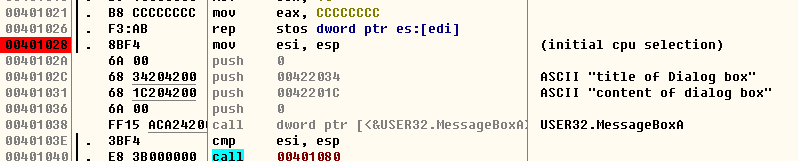


再把得到的可执行文件保存为文件：

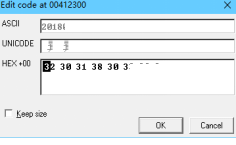
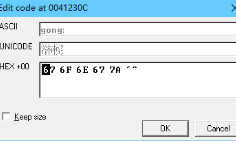


运行保存的文件，没有弹出对话框，也没有报错，说明实验成功。

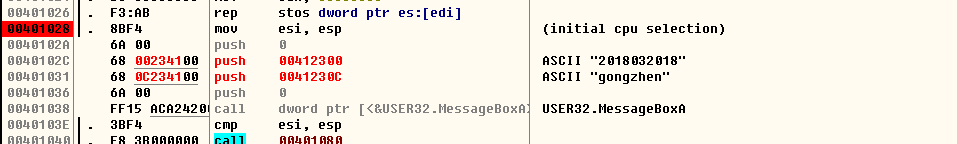
b) 修改对话框中的标题和对话框内的内容，让标题演示自己的学号并让对话框内的内容显示自己姓名的拼音；（要求保存修改以后得到的exe文件）。在上一个实验的基础上，修改程序调用MessageBoxA函数的代码，将两个参数地址改成自己的参数地址：



首先在两个没有代码的内存位置写入自己的信息

然后修改参数的地址



然后用同样的方法保存，运行结果如下

